An Improved Distinguisher for Dragon

Joo Yeon Cho and Josef Pieprzyk

Centre for Advanced Computing – Algorithms and Cryptography,
Department of Computing, Macquarie University,
NSW, Australia, 2109
{jcho,josef}@ics.mq.edu.au

Abstract. Dragon stream cipher is one of the focus ciphers which have reached Phase 2 of the eSTREAM project. In this paper, we present a new method of building a linear distinguisher for Dragon. The distinguisher is constructed by exploiting the biases of two S-boxes and the modular addition which are basic components of the nonlinear function F. The bias of the distinguisher is estimated to be around $2^{-75.32}$ which is better than the bias of the distinguisher presented by Englund and Maximov. We have shown that Dragon is distinguishable from a random cipher by using around $2^{150.6}$ keystream words and 2^{59} memory. In addition, we present a very efficient algorithm for computing the bias of linear approximation of modular addition.
Keywords : Stream Ciphers, eSTREAM, Dragon, Modular Addition.

1 Introduction

Dragon [1] is a word-oriented stream cipher submitted to the eSTREAM project [4]. Dragon is one of the focus ciphers (Software) which are included in Phase 2 of the eSTREAM. During Phase 1, Englund and Maximov presented a distinguishing attack against Dragon [2]. Their distinguisher is constructed using around 2^{155} keystream words and 2^{96} memory.

Unlike Englund and Maximov's work, we use a different approach to find more efficient distinguisher. In a nut shell, we first derive linear approximations for the basic nonlinear blocks used in the cipher, namely, for two S-boxes and for modular additions. Next we combine those approximations and build a linear approximation for the whole state update function F. While combining these elementary approximations we use two basic operations that we call cutting and bypassing. The bypassing operation replaces the original component by its approximation but the cutting operation replaces the original component by zero. Then, we design the distinguisher by linking the approximation of the update function F with the observable output keystream for two specific clocks.

Building the best distinguisher is composed of two steps. First, all the linear masks for the internal approximations are assumed to be identical. Hence, the mask for distinguisher holding the strongest bias can be found easily. Next, the bias of the distinguisher is estimated more accurately by considering the dependencies among internal approximations. This goal is achieved by allowing the different internal masks to approximations that are used to build the distinguisher.

In result, the bias of our distinguisher is estimated to be around $2^{-75.32}$ when 2^{59} bits of internal memory are guessed. Hence, we claim that Dragon is distinguishable from the random process with around $2^{150.6}$ words data complexity and with 2^{59} memory complexity. This complexity is better than the one presented in [2]. Our distinguisher is also described explicitly by showing the best approximations of the nonlinear components of the cipher. In contrast, the previous best attack by Englund and Maximov used a statistical argument to evaluate a bias of the function F.

This paper is organized as follows. Section 2 presents a brief description of Dragon. In Section 3, a series of linear approximations of nonlinear components of Dragon is presented. And then, a distinguisher is built by combining the approximations. In Section 4, the distinguisher is improved by considering the dependencies of intermediate approximations. Section 5 concludes the work.

2 A brief description of Dragon

Dragon consists of a 1024-bit nonlinear feedback register, a nonlinear state update function, and a 64-bit internal memory. Dragon uses two sizes of key and initialization vector that is 128 or 256 bits and produces a 64-bit (two words) output every clock. The nonlinear state update function, which is called the function F, takes six state words (192-bit) as input and produces six words (192 bits) as output. Among the output words of F function, two words are used as new state words and two words are produced as a keystream. The detail structure of the F function is displayed in Figure 1. Suppose that the 32-bit input x is split

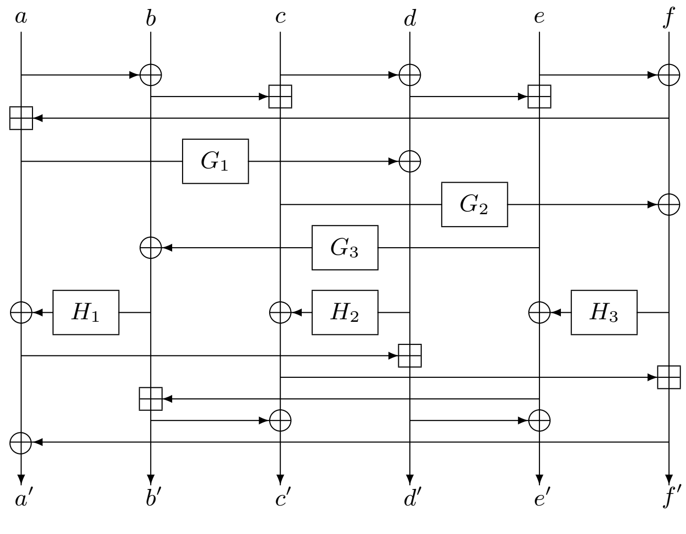

Fig. 1. F function

into four bytes such as $x = x_0||x_1||x_2||x_3$ where x_i denotes a single byte and $||$ denotes a concatenation. Then, the functions G and H that are components of the F function are constructed by using two 8×32 S-boxes, which are called as S_1 and S_2, in the following way.

$$
\begin{aligned}
G_1(x) &= S_1(x_0) \oplus S_1(x_1) \oplus S_1(x_2) \oplus S_2(x_3) \\
G_2(x) &= S_1(x_0) \oplus S_1(x_1) \oplus S_2(x_2) \oplus S_1(x_3) \\
G_3(x) &= S_1(x_0) \oplus S_2(x_1) \oplus S_1(x_2) \oplus S_1(x_3) \\
H_1(x) &= S_2(x_0) \oplus S_2(x_1) \oplus S_2(x_2) \oplus S_1(x_3)
\end{aligned}
$$

$$H_2(x) = S_2(x_0) \oplus S_2(x_1) \oplus S_1(x_2) \oplus S_2(x_3)$$
$$H_3(x) = S_2(x_0) \oplus S_1(x_1) \oplus S_2(x_2) \oplus S_2(x_3)$$

Using the F function, the keystream is generated as follows.

1. Input : $\{B_0, B_1, \ldots, B_{31}\}$ and $M = (M_L||M_R)$
2. $a = B_0, b = B_9, c = B_{16}, d = B_{19}, e = B_{30} \oplus M_L, f = B_{31} \oplus M_R$ where $M = M_R||M_L$.
3. $(a', b', c', d', e', f') = F(a, b, c, d, e, f)$
4. $B_0 = b', B_1 = c'$ and $B_i = B_{i-2}, 2 \leq i \leq 31, M = M + 1$
5. Output : $k = (a'||e')$

For a complete description of Dragon, we refer the reader to the paper [1].

3 A linear distinguisher for Dragon

Let n be a non-negative integer. Given two vectors $x = (x_{n-1}, \ldots, x_0)$ and $y = (y_{n-1}, \ldots, y_0)$ where $x, y \in GF(2^n)$, let $x \cdot y$ denote a standard inner product defined as $x \cdot y = x_{n-1}y_{n-1} \oplus \ldots \oplus x_0y_0$. A linear mask is a constant vector that is used to compute an inner product of a n-bit string.

Let us assume that we have a function $f : \{0,1\}^m \to \{0,1\}^n$ for some positive integers m and n. Given a linear input mask $\Lambda \in GF(2^m)$ and a linear output mask $\Gamma \in GF(2^n)$, the bias of an approximation $\Lambda \cdot x = \Gamma \cdot f(x)$ is measured as follows.

$$\epsilon_f(\Lambda, \Gamma) = 2^{-n}(\#(\Lambda \cdot x \oplus \Gamma \cdot f(x) = 0) - \#(\Lambda \cdot x \oplus \Gamma \cdot f(x) = 1))$$

where $x \in GF(2^m)$ and runs through all possible values. Then, $Pr[\Lambda \cdot x = \Gamma \cdot f(x)] = \frac{1}{2}(1 + \epsilon_f(\Lambda, \Gamma))$. Note that given q independent approximations each having the bias ϵ, the combination of q approximations has the bias of ϵ^q according to the well-known Piling-up Lemma [3].

3.1 Approximations of functions G and H

According to the structure of the functions G and H, the essential components of the functions G and H are the two S-boxes: S_1 and S_2. Hence, the linear approximations of the functions G and H can be constructed by combining approximations of S_1 and S_2 appropriately. In particular, for our distinguisher which will be described next subsection, we need special forms of approximations displayed in Table 1. Note that the approximations of the function G use an input and an output masks which are identical, while those of function H use only an output mask. The reason for this will be explained in Subsection 3.3. We call the approximations of the form $\Gamma \cdot G(x) = \Gamma \cdot x$ **bypassing** approximations, whereas the approximations of the form $\Gamma \cdot H(x) = 0$ **cutting** approximations. Table 1 shows the examples of such approximations that hold high biases.

Approximations of the function H Assume that a 32-bit x is a uniformly distributed random variable. If x is divided into four bytes such as $x = x_0||x_1||x_2||x_3$, where x_i denotes the i-th byte of x, then the approximation $\Gamma \cdot H_1(x) = 0$ can be represented as

$$\Gamma \cdot H_1(x) = \Gamma \cdot S_2(x_0) \oplus \Gamma \cdot S_2(x_1) \oplus \Gamma \cdot S_2(x_2) \oplus \Gamma \cdot S_1(x_3) = 0$$

approximation	bias	example
$\Gamma \cdot H(x) = 0$	$\epsilon_H(0, \Gamma)$	$\epsilon_H(0, 0X4810812B) = -2^{-7.16}$
$\Gamma \cdot x = \Gamma \cdot G_1(x)$	$\epsilon_{G_1}(\Gamma, \Gamma)$	$\epsilon_{G_1}(0x09094102, 0x09094102) = -2^{-9.33}$
$\Gamma \cdot x = \Gamma \cdot G_2(x)$	$\epsilon_{G_2}(\Gamma, \Gamma)$	$\epsilon_{G_2}(0x90904013, 0x90904013) = -2^{-9.81}$

Table 1. Cutting and bypassing approximations of the function G and H

Hence, the bias $\epsilon_{H_1}(0, \Gamma)$ is computed as

$$\epsilon_{H_1}(0, \Gamma) = \epsilon_{S_2}(0, \Gamma)^3 \times \epsilon_{S_1}(0, \Gamma),$$

where $\epsilon_{S_i}(0, \Gamma)$ denotes the bias of the approximation $\Gamma \cdot S_i(x_j) = 0$. Due to the structure of the function H, the approximations $\Gamma \cdot H_1(x) = 0, \Gamma \cdot H_2(x) = 0$ and $\Gamma \cdot H_3(x) = 0$ are isomorphic when the input x is an independent random variable. Hence, $\epsilon_{H_1}(0, \Gamma) = \epsilon_{H_2}(0, \Gamma) = \epsilon_{H_3}(0, \Gamma)$.

Approximations of the function G A 32-bit x is assumed to be a uniformly distributed random variable. If x is divided into four bytes such as $x = x_0||x_1||x_2||x_3$, and a mask Γ is divided into four submasks such as $\Gamma = \Gamma_0||\Gamma_1||\Gamma_2||\Gamma_3$, where $\Gamma_i \in \{0, 1\}^8$, then the approximation $\Gamma \cdot x = \Gamma \cdot G(x)$ can be decomposed into

$$\begin{aligned} \Gamma \cdot (x \oplus G_1(x)) = (\Gamma_0 \cdot x_0 \oplus \Gamma \cdot S_1(x_0)) \oplus (\Gamma_1 \cdot x_1 \oplus \Gamma \cdot S_1(x_1)) \\ \oplus (\Gamma_2 \cdot x_2 \oplus \Gamma \cdot S_1(x_2)) \oplus (\Gamma_3 \cdot x_3 \oplus \Gamma \cdot S_2(x_3)) = 0 \end{aligned}$$

Hence, the bias $\epsilon_G(\Gamma, \Gamma)$ can be computed as follows

$$\epsilon_G(\Gamma, \Gamma) = \epsilon_{S_1(x_0)}(\Gamma_0, \Gamma) \times \epsilon_{S_1(x_1)}(\Gamma_1, \Gamma) \times \epsilon_{S_1(x_2)}(\Gamma_2, \Gamma) \times \epsilon_{S_2(x_3)}(\Gamma_3, \Gamma),$$

where $\epsilon_{S_i(x_j)}(\Gamma, \Gamma_j)$ denotes the bias of the approximation $\Gamma_j \cdot x_j \oplus \Gamma \cdot S_i(x_j) = 0$.

3.2 Linear approximations of modular addition

Let x and y be uniformly distributed random vectors, where $x, y \in GF(2^n)$ for a positive n. Given a mask $\Gamma \in GF(2^n)$, a linear approximation of modular addition where an input and an output masks are Γ is defined as follows:

$$Pr[\Gamma \cdot (x \boxplus y) = \Gamma \cdot (x \oplus y)] = \frac{1}{2}(1 + \epsilon_+(\Gamma, \Gamma)). \tag{1}$$

where the bias of the approximation is denoted by $\epsilon_+(\Gamma, \Gamma)$. Also, given a vector x, the Hamming weight of x is defined as the number of nonzero coordinates of x.

Theorem 1. *Let n and m be positive integers. Given a linear mask $\Gamma = (\gamma_{n-1}, \cdots, \gamma_0)$ where $\gamma_i \in \{0, 1\}$ we assume that the Hamming weight of Γ is m. If a vector $W_\Gamma = (w_{m-1}, w_{m-2} \cdots, w_0)$ denotes the bit positions of Γ where $\gamma_i = 1$, then a bias $\epsilon_+(\Gamma, \Gamma)$ is determined as follows.*
If m is even, then,

$$\epsilon_+(\Gamma, \Gamma) = 2^{-d_1} \quad \textit{where } d_1 = \sum_{i=0}^{m/2-1} (w_{2i+1} - w_{2i}) \tag{2}$$

If m is odd, then

$$\epsilon_+(\Gamma,\Gamma) = 2^{-d_2} \quad \textit{where } d_2 = \sum_{i=1}^{(m-1)/2} (w_{2i} - w_{2i-1}) + w_0 \tag{3}$$

Proof. See Appendix A.

For example, if $\Gamma = 0\text{x}0600018\text{D}$, the Hamming weight of Γ is 7 and $W_\Gamma = (26, 25, 8, 7, 3, 2, 0)$. Hence, $\epsilon_+(\Gamma,\Gamma) = 2^{-[(26-25)+(8-7)+(3-2)]} = 2^{-3}$.

Corollary 1. *Let m be a positive integer. Given a mask vector Γ whose Hamming weight is m, an approximation $\Gamma \cdot (x \boxplus y) = \Gamma \cdot (x \oplus y)$ has at most a bias of $2^{-(m-1)/2}$.*

Proof. See Appendix B.

3.3 Linear approximation of F function

According to the state update rule of Dragon, there is the following relation between two state words chosen at clock t and $t+15$. [1]

$$B_0[t] = B_{30}[t+15], \quad t = 0, 1, \ldots \tag{4}$$

We know that $a = B_0$ and $e = B_{30} \oplus M_L$ where a and e are two words out of six input words of the F function. Then, we try to find the linear approximations $\Gamma \cdot a' = \Gamma \cdot a$ and $\Gamma \cdot e' = \Gamma \cdot e$ where a' and e' are two output words of the F function that are produced as a keystream.

We regard the outputs of the functions G and H as independent and uniformly distributed random variables. This assumption is reasonable since each G or H function has unique input parameters at given clock t so that the output of the functions G and H are mutually independent. Hence, the functions G and H are described without input parameters in the following expressions.

The approximation of a' As illustrated in Figure 1, an output word a' is expressed as follows.

$$a' = [(a \boxplus (e \oplus f)) \oplus H_1] \oplus [(e \oplus f \oplus G_2) \boxplus (H_2 \oplus ((a \oplus b) \boxplus c))] \tag{5}$$

Due to the linear property of Γ, we know that

$$\Gamma \cdot a' = \Gamma \cdot [(a \boxplus (e \oplus f)) \oplus H_1] \oplus \Gamma \cdot [(e \oplus f \oplus G_2) \boxplus (H_2 \oplus ((a \oplus b) \boxplus c))].$$

By applying Approximation (1), we get

$$\Gamma \cdot [(e \oplus f \oplus G_2) \boxplus (H_2 \oplus ((a \oplus b) \boxplus c))] = \Gamma \cdot (e \oplus f \oplus G_2) \oplus \Gamma \cdot [(H_2 \oplus ((a \oplus b) \boxplus c))]$$

which holds with the bias of $\epsilon_+(\Gamma,\Gamma)$. Hence, we have

$$\Gamma \cdot a' = \Gamma \cdot [(a \boxplus (e \oplus f)) \oplus H_1] \oplus \Gamma \cdot (e \oplus f \oplus G_2) \oplus \Gamma \cdot [H_2 \oplus ((a \oplus b) \boxplus c)].$$

Next, the two types of approximations are used in our analysis: First, cutting approximations are used for the functions H_1 and H_2. That is, we use $\Gamma \cdot H_1 = 0$ and $\Gamma \cdot H_2 = 0$ which hold

[1] This relation was also observed in [2].

the biases of $\epsilon_{H_1}(0, \Gamma)$ and $\epsilon_{H_2}(0, \Gamma)$, respectively. Intuitively, these approximations allow to simplify the form of the final approximation of the function F by replacing the output variables of a nonlinear component by zeros.

Second, bypassing approximations are used for the function G_2. That is, we use $\Gamma \cdot G_2 = \Gamma \cdot [(a \oplus b) \boxplus c]$ that has a bias of $\epsilon_{G_2}(\Gamma, \Gamma)$. In this category of approximations we are able to replace a combination of output variables by the same combination of input variables. Then, we have

$$\begin{aligned}\Gamma \cdot a' &= \Gamma \cdot [(a \boxplus (e \oplus f))] \oplus \Gamma \cdot (e \oplus f \oplus [(a \oplus b) \boxplus c]) \oplus \Gamma \cdot [(a \oplus b) \boxplus c] \\ &= \Gamma \cdot [(a \boxplus (e \oplus f))] \oplus \Gamma \cdot (e \oplus f)\end{aligned}$$

Finally, by applying Approximation (1) for the modular addition, we obtain

$$\Gamma \cdot a' = \Gamma \cdot a \tag{6}$$

We know that $\Gamma \cdot [(a \boxplus (e \oplus f))] = \Gamma \cdot a \oplus \Gamma \cdot (e \oplus f)$ holds the bias of $\epsilon_+(\Gamma, \Gamma)$. Therefore, the bias of Approximation (6) can be computed from the biases of the component approximations as follows:

$$\epsilon_{a'}(\Gamma, \Gamma) = \epsilon_+(\Gamma, \Gamma)^2 \times \epsilon_{H_1}(0, \Gamma) \times \epsilon_{H_2}(0, \Gamma) \times \epsilon_{G_2}(\Gamma, \Gamma).$$

Since the 32-bit word a' is an upper part of a 64-bit keystream output at each clock, Approximation (6) is equivalent to the following expression.

$$\Gamma \cdot k_0[t] = \Gamma \cdot B_0[t] \tag{7}$$

where $k_0[t]$ denotes the upper part of a 64-bit k at clock t.

The approximation of e' As depicted in Figure 1, an output word e' is described as

$$e' = [((a \boxplus (e \oplus f)) \oplus H_1) \boxplus (c \oplus d \oplus G_1)] \oplus [H_3 \oplus ((c \oplus d) \boxplus e)] \tag{8}$$

Similarly to the case of a', we would like to obtain an approximation $\Gamma \cdot e' = \Gamma \cdot e$. To do this, we first apply Approximation (1) for modular addition and as the result we get

$$\Gamma \cdot e' = \Gamma \cdot [(a \boxplus (e \oplus f)) \oplus H_1] \oplus \Gamma \cdot (c \oplus d \oplus G_1) \oplus \Gamma \cdot [H_3 \oplus ((c \oplus d) \boxplus e)]$$

Next, we apply the cutting approximations for functions H_1, H_3 and the bypassing approximation for the function G_1. That is, we use the following approximations

$$\Gamma \cdot H_1 = 0, \quad \Gamma \cdot H_3 = 0, \quad \Gamma \cdot G_1 = \Gamma \cdot [a \boxplus (e \oplus f)]$$

that hold with the biases of $\epsilon_{H_1}(0, \Gamma), \epsilon_{H_3}(0, \Gamma)$ and $\epsilon_{G_1}(\Gamma, \Gamma)$, respectively. These approximations are plugged into the above relation and we obtain the following result

$$\begin{aligned}\Gamma \cdot e' &= \Gamma \cdot [(a \boxplus (e \oplus f))] \oplus \Gamma \cdot (c \oplus d \oplus [a \boxplus (e \oplus f)]) \oplus \Gamma \cdot [(c \oplus d) \boxplus e] \\ &= \Gamma \cdot (c \oplus d) \oplus \Gamma \cdot [(c \oplus d) \boxplus e].\end{aligned}$$

Finally, by applying Approximation (1) for modular addition, we can conclude that output e' and input e satisfy the following approximation

$$\Gamma \cdot e' = \Gamma \cdot e \tag{9}$$

with the bias of $\epsilon_{e'}(\Gamma, \Gamma) = \epsilon_+(\Gamma, \Gamma)^2 \times \epsilon_{H_1}(0, \Gamma) \times \epsilon_{H_3}(0, \Gamma) \times \epsilon_{G_1}(\Gamma, \Gamma)$. Since the 32-bit word e' is a lower part of a 64-bit keystream output k at each clock, Approximation (9) is equivalent to the following expression.

$$\Gamma \cdot k_1[t] = \Gamma \cdot (B_{30}[t] \oplus M_L[t]) \tag{10}$$

where $k_1[t]$ and $M_L[t]$ denote the lower part of a 64-bit k and the upper part of a 64-bit memory word M at clock t, respectively.

3.4 Building the distinguisher

According to Equation (4), Approximations (7) and (10) can be combined in such a way that

$$\Gamma \cdot k_0[t] = \Gamma \cdot B_0[t] = \Gamma \cdot B_{30}[t+15] = \Gamma \cdot (k_1[t+15] \oplus M_L[t+15])$$

By guessing (partially) the initial value of M, we can build the following distinguisher.

$$\Gamma \cdot k_0[t] = \Gamma \cdot (k_1[t+15]) \tag{11}$$

For the correctly guessed initial value of M, the distinguisher (11) shows the bias of

$$\begin{aligned}\epsilon_D(\Gamma,\Gamma) &= \epsilon_{a'}(\Gamma,\Gamma) \times \epsilon_{e'}(\Gamma,\Gamma) \\ &= \epsilon_+(\Gamma,\Gamma)^4 \times \epsilon_{H_1}(0,\Gamma)^2 \times \epsilon_{H_2}(0,\Gamma) \times \epsilon_{H_3}(0,\Gamma) \times \epsilon_{G_1}(\Gamma,\Gamma) \times \epsilon_{G_2}(\Gamma,\Gamma) \quad (12)\end{aligned}$$

We implemented a mask search for the function F to achieve the distinguisher with the biggest bias. The space of a linear mask Γ contains $2^{32}-1$ elements. For each mask Γ, the following procedure is performed to compute the bias given by Expression (12).

Step 1. For an input x that varies from 0 to 255, measure the biases of $\Gamma \cdot S_1(x) = 0$ and $\Gamma \cdot S_2(x) = 0$, respectively. Then, compute $\epsilon_{H_1}(0,\Gamma)$, $\epsilon_{H_2}(0,\Gamma)$ and $\epsilon_{H_3}(0,\Gamma)$.
Step 2. The mask Γ is divided into four submasks $\Gamma = \Gamma_0||\Gamma_1||\Gamma_2||\Gamma_3$. For an input x that varies from 0 to 255, measure the bias of $\Gamma \cdot S_1(x) = \Gamma_i \cdot x$ and $\Gamma \cdot S_2(x) = \Gamma_i \cdot x$ for some $0 \le i \le 3$. Then, compute the biases $\epsilon_{G_1}(\Gamma,\Gamma)$ and $\epsilon_{G_2}(\Gamma,\Gamma)$.
Step 3. Compute $\epsilon_+(\Gamma,\Gamma)$ using Theorem 1.
Step 4. Finally, compute $\epsilon_D(\Gamma,\Gamma)$.

3.5 Our results

We searched for a linear mask which maximizes the bias (12). Due to Corollary 1, the bias $\epsilon_+(\Gamma,\Gamma)$ decreases exponentially by the increment of the Hamming weight of a linear mask. Hence, there is a better chance to achieve higher bias when the Hamming weight is smaller. In result, we found that the best linear approximation of the function F is using the distinguisher (11) with the mask Γ = 0x0600018D. The bias of the distinguisher in this case is $2^{-75.8}$ as listed in Table 2. In order to remove the impact of the unknown state of the internal memory on the bias, we need to guess the first 27 bits of initial value of M_L and 32 bits of M_R. Hence, we need to store all possible values of the internal state which takes $2^{27+32} = 2^{59}$ bits.

Γ	$\epsilon_+(0,\Gamma)$	$\epsilon_H(\Gamma,\Gamma)$	$\epsilon_{G_1}(\Gamma,\Gamma)$	$\epsilon_{G_2}(\Gamma,\Gamma)$	$\epsilon_{a'}(\Gamma,\Gamma)$	$\epsilon_{e'}(\Gamma,\Gamma)$	$\epsilon_D(\Gamma,\Gamma)$
0x0600018D	2^{-3}	$-2^{-8.58}$	$2^{-13.59}$	$2^{-15.91}$	$-2^{-39.1}$	$-2^{-36.7}$	$2^{-75.8}$

Table 2. The bias of distinguisher

4 Improving the distinguisher

In this section, we generalize a method presented in Section 3. [2] First, we apply different linear masks for each component of the function F and combine them to build the distinguisher. Figure 2 illustrates how different linear masks can be applied for each component of the function F. Second, we consider the internal dependencies for the approximations of the

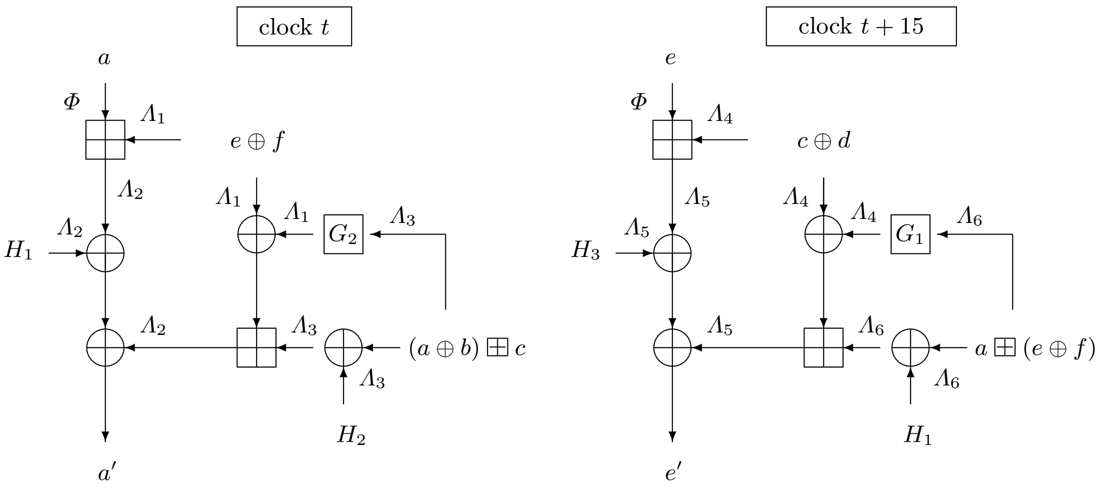

Fig. 2. Generalized linear masks for approximations of the function F

function F. Since the approximations of the components of the F are internally canceled, the correlation of the distinguisher can be accurately computed by exploiting all possible internal approximations induced by different linear masks. Based on these two observations, we searched extensively for a new distinguisher that could improve the efficiency of attack. A new distinguisher can be built from the relation of (a, a') and (e, e') presented in Equations (5) and (8). A basic requirement for establishing a distinguisher is to apply the identical mask Φ to the state a at clock t and to the state e at clock $t + 15$, as stated in Section 3. However, this time, the other internal masks can be different, as shown in Figure 2. We set up the six masks, $\{\Lambda_1, \cdots, \Lambda_6\}$, for the components of the F and build the following approximations:

$$\begin{aligned}
\Lambda_2 \cdot a' &= \Lambda_2 \cdot [(a \boxplus (e \oplus f)) \oplus H_1] \oplus \Lambda_2 \cdot [(e \oplus f \oplus G_2) \boxplus (H_2 \oplus ((a \oplus b) \boxplus c))] \\
&= \Phi \cdot a \oplus \Lambda_1 \cdot (e \oplus f) \oplus \Lambda_2 \cdot H_1 \oplus \Lambda_1 \cdot (e \oplus f \oplus G_2) \oplus \Lambda_3 \cdot [H_2 \oplus ((a \oplus b) \boxplus c)] \\
&= \Phi \cdot a \qquad (13)
\end{aligned}$$

$$\begin{aligned}
\Lambda_5 \cdot e' &= \Lambda_5 \cdot [((c \oplus d) \boxplus e) \oplus H_3] \oplus \Lambda_5 \cdot [((a \boxplus (e \oplus f)) \oplus H_1) \boxplus (c \oplus d \oplus G_1)] \\
&= \Lambda_4 \cdot (c \oplus d) \oplus \Phi \cdot e \oplus \Lambda_5 \cdot H_3 \oplus \Lambda_6 \cdot (a \boxplus (e \oplus f)) \oplus \Lambda_6 \cdot H_1 \oplus \Lambda_4 \cdot (c \oplus d \oplus G_1) \\
&= \Phi \cdot e \qquad (14)
\end{aligned}$$

[2] This section was inspired by the distinguishing attack on SNOW 2.0 presented by Nyberg and Wallen [6].

The component-wise approximations required for Approximations (13) and (14) are listed in Tables 3 and 4. According to the well-known theorem [5] the correlation of approximations

approximation	bias
$\Phi \cdot x \oplus \Lambda_1 \cdot y \oplus \Lambda_2 \cdot (x \boxplus y) = 0$	$\epsilon_+(\Phi, \Lambda_1, \Lambda_2)$
$\Lambda_2 \cdot H_1 = 0$	$\epsilon_{H_1}(0, \Lambda_2)$
$\Lambda_3 \cdot H_2 = 0$	$\epsilon_{H_2}(0, \Lambda_3)$
$\Lambda_3 \cdot x \oplus \Lambda_1 \cdot G_1(x) = 0$	$\epsilon_{G_1}(\Lambda_3, \Lambda_1)$
$\Lambda_1 \cdot x \oplus \Lambda_3 \cdot y \oplus \Lambda_2 \cdot (x \boxplus y) = 0$	$\epsilon_+(\Lambda_1, \Lambda_3, \Lambda_2)$

Table 3. Component approximations for Equation (13)

approximation	bias
$\Phi \cdot x \oplus \Lambda_4 \cdot y \oplus \Lambda_5 \cdot (x \boxplus y) = 0$	$\epsilon_+(\Phi, \Lambda_4, \Lambda_5)$
$\Lambda_5 \cdot H_3 = 0$	$\epsilon_{H_3}(0, \Lambda_5)$
$\Lambda_6 \cdot H_1 = 0$	$\epsilon_{H_1}(0, \Lambda_6)$
$\Lambda_6 \cdot x \oplus \Lambda_4 \cdot G_2(x) = 0$	$\epsilon_{G_2}(\Lambda_6, \Lambda_4)$
$\Lambda_4 \cdot x \oplus \Lambda_6 \cdot y \oplus \Lambda_5 \cdot (x \boxplus y) = 0$	$\epsilon_+(\Lambda_4, \Lambda_6, \Lambda_5)$

Table 4. Component approximations for Equation (14)

can be computed as a sum of partial correlations over all intermediate linear masks. For theoretical analysis of the theorem, we refer the reader to the paper of [5]. Hence, the bias of Approximation (13) is computed as a sum of partial biases induced by the masks of Λ_1, Λ_2 and Λ_3 as follows:

$$\epsilon_{a'}(\Phi, \Lambda_2) = \epsilon_{H_1}(0, \Lambda_2) \sum_{\Lambda_1} \epsilon_+(\Phi, \Lambda_1, \Lambda_2) \sum_{\Lambda_3} \epsilon_+(\Lambda_3, \Lambda_1, \Lambda_2) \epsilon_{G_2}(\Lambda_3, \Lambda_1) \epsilon_{H_2}(0, \Lambda_3) \tag{15}$$

Similarly, the bias of Approximation (14) using the masks of Λ_4, Λ_5 and Λ_6 can be computed as follows:

$$\epsilon_{e'}(\Phi, \Lambda_5) = \epsilon_{H_3}(0, \Lambda_5) \sum_{\Lambda_4} \epsilon_+(\Phi, \Lambda_4, \Lambda_5) \sum_{\Lambda_6} \epsilon_+(\Lambda_4, \Lambda_6, \Lambda_5) \epsilon_{G_1}(\Lambda_6, \Lambda_4) \epsilon_{H_1}(0, \Lambda_6) \tag{16}$$

Hence, according to Subsection 3.4, a new distinguisher can be derived from Approximations (13) and (14) as follows:

$$\Lambda_2 \cdot k_0[t] = \Lambda_5 \cdot k_1[t + 15] \tag{17}$$

with the bias of

$$\epsilon_D(\Lambda_2, \Lambda_5) = \sum_{\Phi} \epsilon_{a'}(\Phi, \Lambda_2) \epsilon_{e'}(\Phi, \Lambda_5). \tag{18}$$

4.1 Experiments

In order to find the distinguisher holding the biggest bias, we need to search all possible combinations of Γ_2 and Γ_5 and test their biases by Equation (18). Furthermore, for each Γ_2

and Γ_5, the computation of Equation (18) requires a large number of iterations due to the space that all the intermediate masks have. Hence, our experiments focus on reducing the overall space of the masks that are necessarily required for the computation of the bias of the distinguisher. To achieve this goal, we implemented two techniques that can remove a large portion of terms from the summation involved in Equation (18).

First technique is to remove, from Equation (15) and (16), the unnecessary terms caused by the condition of $\epsilon_+ = 0$. The approximations of the modular addition have non-trivial biases only in a portion of space which is determined by the values of an input and an output masks.

Lemma 1. *Assume that the bias of the approximation $\Lambda_1 \cdot x \oplus \Lambda_3 \cdot y \oplus \Lambda_2 \cdot (x \boxplus y) = 0$ is represented by $\epsilon_+(\Lambda_1, \Lambda_3, \Lambda_2)$. Given $\Lambda_2 = b_{31}b_{30}\cdots b_0$ where b_i stands for the i-th bit of Λ_2, we assume that the most significant non-zero bit of Λ_2 is located in the bit position of b_t where $0 \leq t \leq 31$. Then, the bias $\epsilon_+(\Lambda_1, \Lambda_3, \Lambda_2)$ is zero when $\Lambda_1, \Lambda_3 < 2^t$ or $\Lambda_1, \Lambda_3 \geq 2^{t+1}$. In other words, we see that*

$$\sum_{\Lambda_1=1}^{2^{32}-1} \sum_{\Lambda_3=1}^{2^{32}-1} \epsilon_+(\Lambda_1, \Lambda_3, \Lambda_2) = \sum_{\Lambda_1=2^t}^{2^{t+1}-1} \sum_{\Lambda_3=2^t}^{2^{t+1}-1} \epsilon_+(\Lambda_1, \Lambda_3, \Lambda_2)$$

Proof. See Appendix C.

According to Lemma 1, the iteration range of the biases (15) and (16) depends on the space of the output mask of the modular addition. For example, if $\Lambda_2 = \mathrm{0x0600018D}$, then, $\epsilon_+(\Lambda_1, \Lambda_3, \Lambda_2)$ becomes zero when $\Lambda_1, \Lambda_3 < \mathrm{0x04000000}$ or $\Lambda_1, \Lambda_3 \geq \mathrm{0x08000000}$.

Next technique is to restrict, by some value, the correlations of the modular additions used in Equation (15) and (16) and to reduce the number of iterations. Namely, instead of iterating the full space of the intermediate masks, we use only relatively highly biased approximations of the modular additions for estimating the bias of the distinguisher. In the paper of [6], authors proposed an efficient algorithm for finding all input and output masks for addition with a given correlation. This algorithm enables us to reduce the number of iteration for Equation (15) and (16) significantly. We restricted the effective correlation of the modular addition up to $\pm 2^{-24}$, as suggested in [6].

Based on these techniques, we re-calculated the bias of the Distinguisher (17) and found that the bias is estimated to be $2^{-75.32}$. It is interesting to observe how the biases can be improved by considering the dependencies of the combinations of the approximation. Table 5 shows that the bias measurement without considering the dependencies can underestimate the real bias of the approximation.

Due to the restrictions on computing resources, we searched the best distinguisher under the condition that $\Lambda_2 = \Lambda_5$ and we could not perform the experiment for the cases when the different values of Λ_2 and Λ_5 are allowed. Even though Bias (18) tends to be high when $\Lambda_2 = \Lambda_5$, there is a possibility that two different values of Λ_2 and Λ_5 may lead to a distinguisher with a bigger bias. We leave this issue as an open problem.

5 Conclusion

In this paper, we presented a new distinguisher for Dragon. Since the amount of observations for the distinguishing attack is by far larger than the limit of keystream available from a single key, our distinguisher leads only to a theoretical attack on Dragon. However, our

$\Lambda_2 = \Lambda_5$	ϵ_D without dependencies	ϵ_D with dependencies
0x0600018D	$2^{-75.81}$	$2^{-75.32}$
0x002C0039	$-2^{-129.55}$	$2^{-81.77}$
0x00001809	$2^{-84.13}$	$2^{-79.51}$

Table 5. Comparison of the bias of the distinguisher computed by two methods

analysis shows that some approximations of the functions G and H have larger biases than the ones expected by the designers. As far as we know, our distinguisher is the best one for Dragon published so far in open literature. In addition, we present an efficient algorithm to compute the bias of approximation of modular addition, which is expected to be useful for other attacks against ciphers using modular additions.

Acknowledgment We wish to thank Matt Henricksen for invaluable comments. The authors were supported by ARC grants DP0451484, DP0663452 and Macquarie University ARC Safety Net Grant.

References

1. E. Dawson, K. Chen, M. Henricksen, W. Millan, L. Simpson, H. Lee, and S. Moon, *Dragon: A fast word based stream cipher*, eSTREAM, ECRYPT Stream Cipher Project, Report 2005/006, 2005, http://www.ecrypt.eu.org/stream.
2. H. Englund and A. Maximov, *Attack the Dragon.*, Progress in Cryptology - INDOCRYPT 2005, Lecture Notes in Computer Science, vol. 3797, Springer, 2005, pp. 130–142.
3. M. Matsui, *Linear cryptoanalysis method for DES cipher.*, Advances in Cryptology - EUROCRYPT '93, Lecture Notes in Computer Science, vol. 765, Springer, 1993, pp. 386–397.
4. ECRYPT NoE, *eSTREAM - the ECRYPT stream cipher project*, Available at http://www.ecrypt.eu.org/stream/, 2005.
5. K. Nyberg, *Correlation theorems in cryptanalysis*, Discrete Applied Mathematics **111** (2001), no. 1-2, 177–188.
6. K. Nyberg and J. Wallen, *Improved linear distinguishers for SNOW 2.0.*, Fast Software Encryption - FSE 2006, Lecture Notes in Computer Science, vol. 4047, Springer, 2006, pp. 144–162.

A Proof of Theorem 1

Suppose that $z = x \boxplus y$ where $x = (x_{n-1}, \cdots, x_0), y = (y_{n-1}, \cdots, y_0)$ and $z = (z_{n-1}, \cdots, z_0)$. Then, each z_i bit is expressed a function of $x_i, \cdots, x_0$ and $y_i, \cdots, y_0$ bits as follows.

$$z_0 = x_0 \oplus y_0, \quad z_i = x_i \oplus y_i \oplus x_{i-1}y_{i-1} \oplus \sum_{j=0}^{i-2} x_j y_j \prod_{k=j+1}^{i-1} (x_k \oplus y_k), \quad i = 1, \cdots, n$$

If we define the carry $R(x, y)$ as

$$R(x,y)_0 = x_0 y_0, \quad R(x,y)_i = x_i y_i \oplus \sum_{j=0}^{(i-1)} x_j y_j \prod_{k=j+1}^{i} (x_k \oplus y_k), \quad i = 1, 2, \ldots$$

Then, it is clear that $z_i = x_i \oplus y_i \oplus R(x,y)_{i-1}$ for $i > 0$. By the definition, $R(x,y)_i$ has the following recursive relation.

$$R(x,y)_i = x_i y_i \oplus (x_i \oplus y_i) R(x,y)_{i-1} \tag{19}$$

First, we examine the bias of the Γ of which the Hamming weight is 2, i.e. $m = 2$. Without loss of generality, we assume that $\gamma_i = 1$ and $\gamma_j = 1$ where $0 \le j < i < n$. Then, by Relation (19), Approximation (1) is expressed as

$$\begin{aligned}\Gamma \cdot (x \boxplus y) \oplus \Gamma \cdot (x \oplus y) &= z_i \oplus z_j \oplus (x_i \oplus y_i) \oplus (x_j \oplus y_j)\\ &= R(x,y)_{i-1} \oplus R(x,y)_{j-1}\\ &= x_{i-1}y_{i-1} \oplus (x_{i-1} \oplus y_{i-1}) R(x,y)_{i-2} \oplus R(x,y)_{j-1}\end{aligned}$$

Let us denote $p_{i-1} = Pr[R(x,y)_{i-1} \oplus R(x,y)_{j-1} = 0]$. Since x_i and y_i are assumed as uniformly distributed random variables, the probability p_{i-1} is split into the three cases as follows.

$$p_{i-1} = \begin{cases} Pr[R(x,y)_{j-1} = 0], & \text{if } (x_{i-1}, y_{i-1}) = (0,0)\\ Pr[1 \oplus R(x,y)_{j-1} = 0], & \text{if } (x_{i-1}, y_{i-1}) = (1,1)\\ Pr[R(x,y)_{i-2} \oplus R(x,y)_{j-1} = 0], & \text{if } (x_{i-1}, y_{i-1}) = (0,1),(1,0)\end{cases}$$

Clearly, $Pr[R(x,y)_{j-1} = 0] = 1 - Pr[1 \oplus R(x,y)_{j-1} = 0]$. Hence, we get

$$p_{i-1} = \frac{1}{4} + \frac{1}{2} Pr[R(x,y)_{i-2} \oplus R(x,y)_{j-1} = 0] = \frac{1}{4} + \frac{1}{2} p_{i-2}$$

If $j = i-1$, then $Pr[R(x,y)_{i-2} \oplus R(x,y)_{j-1} = 0] = 1$. Hence, $p_{i-1} = \frac{1}{4} + \frac{1}{2} = \frac{3}{4}$. Otherwise, p_{i-2} is determined recursively by the same technique used as above until p_{j-1} is reached. Hence, we obtain the following result.

$$p_{i-1} = \frac{1}{4}(1 + \cdots + 2^{-(i-j-1)}) + 2^{-(i-j)} = \frac{1}{2}(1 + 2^{-(i-j)}) \tag{20}$$

Therefore, the $\epsilon_+(\Gamma, \Gamma)$ is determined by only the difference between two position i and j of Γ.

Next, we consider the case that Γ has an arbitrary Hamming weight, which is denoted m. Assume that we convert m into an even number m' by using the following technique.

- If m is even, then set $m' = m$.
- If m is odd and $\gamma_0 = 0$, then set $\gamma_0 = 1$ and $m' = m + 1$.
- If m is odd and $\gamma_0 = 1$, then set $\gamma_0 = 0$ and $m' = m - 1$.

In result, the Γ is transformed to Γ' which has the Hamming weight of m'. Since the modular addition is linear for the least significant bit, $\epsilon_+(\Gamma, \Gamma) = \epsilon_+(\Gamma', \Gamma')$. Hence, a new position vector for Γ' is defined as $W_{\Gamma'} = (w_{m'-1}, \cdots, w_0)$ where $0 \le w_j \le n$.

Now, we decompose Γ' into a combination of sub-masks which have the Hamming weight of 2. That is, Γ is expressed as

$$\Gamma = \Omega_{m'/2-1} \oplus \cdots \oplus \Omega_0$$

where Ω_k is a sub-mask which has the nonzero coordinates only at position w_{2k} and w_{2k+1} for $k = 0, 1, \cdots, \frac{m'}{2} - 1$. Clearly, the number of such sub-masks is $\frac{m'}{2}$. For example, if $\Gamma = (0,0,1,1,0,1,1)$, then $\Gamma = \Omega_1 \oplus \Omega_0 = (0,0,1,1,0,0,0) \oplus (0,0,0,0,0,1,1)$.

From (20), we know that the bias of $\Omega_k \cdot (x \boxplus y) \oplus \Omega_k \cdot (x \oplus y)$ is only determined by the difference $w_{2k+1} - w_{2k}$. Hence, according to the Piling-up Lemma [3], the bias of $\Gamma \cdot (x \boxplus y) \oplus \Gamma \cdot (x \oplus y)$ is obtained by combining the $\frac{m'}{2}$ approximations independently. Note that the there are no inter-dependencies among sub-masks. Therefore, the claimed bias is computed as

$$\epsilon_+(\Gamma, \Gamma) = 2^{-[w_{m'-1} - w_{m'-2}] + \cdots + [w_1 - w_0]}$$

If m' is replaced by m, we obtain the claimed bias. $\square$

B Proof of Corollary 1

Recall Theorem 1. If m is even, then,

$$d_1 = \sum_{i=0}^{m/2-1} (w_{2i+1} - w_{2i}) \geq \sum_{i=0}^{m/2-1} 1 = m/2$$

If m is odd, then,

$$d_2 = \sum_{i=1}^{(m-1)/2} (w_{2i} - w_{2i-1}) + w_0 \geq \sum_{i=1}^{(m-1)/2} 1 = (m-1)/2$$

Hence, the bias $\epsilon_+(\Gamma, \Gamma) \leq 2^{-(m-1)/2}$. $\square$

C Proof of Lemma 1

Let x_i and y_i denote the i-th bits of 32-bit words x and y. According to the notation used in Appendix A, the approximation using the output mask Λ_2 can be expressed as

$$\Lambda_2 \cdot (x \boxplus y) = x_t \oplus y_t \oplus R(x, y)_{t-1} \oplus A(x, y)_{t-1}$$

where $A(x, y)_{t-1}$ is a function which does not contain x_t and y_t bits as variables.

When $\Lambda_1 < 2^t$ or $\Lambda_3 < 2^t$, the input approximation $\Lambda_1 \cdot x \oplus \Lambda_3 \cdot y$ does not contain x_t or y_t bit as a variable. Thus, $\Lambda_1 \cdot x \oplus \Lambda_3 \cdot y \oplus \Lambda_2 \cdot (x \boxplus y)$ retains a linear term x_t or y_t so that the bias of the approximation becomes zero.

On the other hand, given $\Lambda_1 \geq 2^{t+1}$ or $\Lambda_3 \geq 2^{t+1}$, the input approximation $\Lambda_1 \cdot x \oplus \Lambda_3 \cdot y$ contain x_u or y_v bit as a variable where $u, v > t$. Thus, $\Lambda_1 \cdot x \oplus \Lambda_3 \cdot y \oplus \Lambda_2 \cdot (x \boxplus y)$ retains a linear term x_u or y_v so that the bias of the approximation becomes zero. $\square$